Méthodologie de la programmation

Cours no. 6

Jean-Sébastien Coron

coron@clipper.ens.fr

Université du Luxembourg

Plan du cours

- Les fonctions en C.
- Programmation structurée.

Programmation structurée

- Quand la taille d'un programme augmente, il faut le découper en modules indépendants plus petit.
  - Cela rend le programme plus lisible.
  - Cela facilite la recherche des erreurs, module par module.
  - On peut réutiliser le même module plusieurs fois dans le programme.
- Fonction en C:
  - Permet ce découpage en module.
  - Prend en entrée des arguments et renvoie une sortie.

Fonction

- Toute fonction a un prototype et un corps.
- prototype de la fonction:
  - `int factoriel(int n)`
  - `double max(double a,double b)`
  - Donne le "mode d'emploi" de la fonction:
    - ✓ Le nom de la fonction.
    - ✓ Les arguments en entrée et leur type.
    - ✓ Le type retourné par la fonction.
- Corps de la fonction:
  - Déclaration de variables locales.
  - Instructions à exécuter.
  - Variable à retourner.

Exemple

```
double max(double a,double b)
{
  double m;
  if(a>b)
  {
    m=a;
  }
  else
  {
    m=b;
  }
  return m;
}
```

Prototype de la fonction

- `double max(double a,double b)`
  - `max` est le nom de la fonction.
  - `double a` est le premier argument.
  - `double b` est le deuxième argument.
  - `double` est le type de retour.

Utilisation de la fonction

- Pour la fonction ayant le prototype
  `double max(double a,double b)`
  - Soient `x,y,z` des variables de type `double`.
  - Alors l'instruction

    ```
    z=max(x,y);
    ```

    applique la fonction `max` aux variables `x` et `y`.
  - et stocke le résultat dans `z`.

Que se passe-t'il ?

- Soit le prototype:
  - `double max(double a,double b)`
- Lors de l'exécution de l'instruction:
  - `z=max(x,y);`
  - Le programme copie le contenu de la variable `x` dans `a`.
  - Le programme copie le contenu de la variable `y` dans `b`.
  - Le programme exécute les instructions contenues dans la fonction `max`.
  - Le programme stocke la valeur retournée par la fonction dans `z`.

Typage

- Il faut que la fonction soit appelée avec des variables ayant le type correspondant au prototype de la fonction:
  - `double max(double a,double b)`
  - `z=max(x,y);`
  - Il faut que `x` et `y` soient de type `double`.
- Il faut que la valeur retournée par la fonction soit stockée dans une variable du même type.
  - If faut que `z` soit de type `double`.

Exemple

```
#include <stdio.h>
double max(double a,double b)
{
  double m;
  if(a>b) m=a; else m=b;
  return m;
}
int main()
{
  double x=3.5;
  double y=3.2;
  double z=max(x,y);
}
```

Fonctions `void`

- Une fonction de type `void` est une fonction qui ne retourne aucune valeur.

```
#include <stdio.h>
void affiche(int a)
{
  printf("La valeur est:%d\n",a);
}

int main()
{
  int u=3;
  affiche(u);
}
```

Affichage d'un tableau

```
#include <stdio.h>
void affiche(int tab[],int n)
{
  int i;
  for(i=0;i<n;i++) printf("%d ",tab[i]);
  printf("\n");
}

int main()
{
  int t[5]={1,3,6,5,1}
  affiche(t,5);
}
```

Variables locales/globales

- Variable locale.
  - Variable déclarée à l'intérieur d'une fonction.
  - N'est visible qu'à l'intérieur de la fonction.
- Variable globale.
  - Variable déclarée à l'extérieure de toute fonction
  - Visibles dans toutes les fonctions.

Variables locales/globales

```
#include <stdio.h>
int u=0;
void additionne(int a)
{
  u=u+a;
}
int main()
{
  printf("u=%d\n",u);  // u=0
  additionne(5);
  printf("u=%d\n",u);  // u=5
}
```

Passage par valeur

- Lors de l'appel d'une fonction, les valeur des variables données en argument sont copiées dans les variables paramêtres de la fonction.
  - `double max(double a,double b)`
  - `z=max(x,y);`
  - Le contenu des variables `x` et `y` est copié dans `a` et `b`.
- Passage par valeur:
  - Si le contenu des variables `a` ou `b` est modifé à l'intérieur de la fonction, cela n'affecte pas le contenu des variables `x` et `y`.

Passage par valeur

```
#include <stdio.h>
int incremente(int a)
{
  a=a+1;
  return a;
}
int main()
{
  int u=5;
  int v;
  v=incremente(u);
  printf("u=%d\n",u); // u=5
  printf("v=%d\n",v); // v=6
}
```